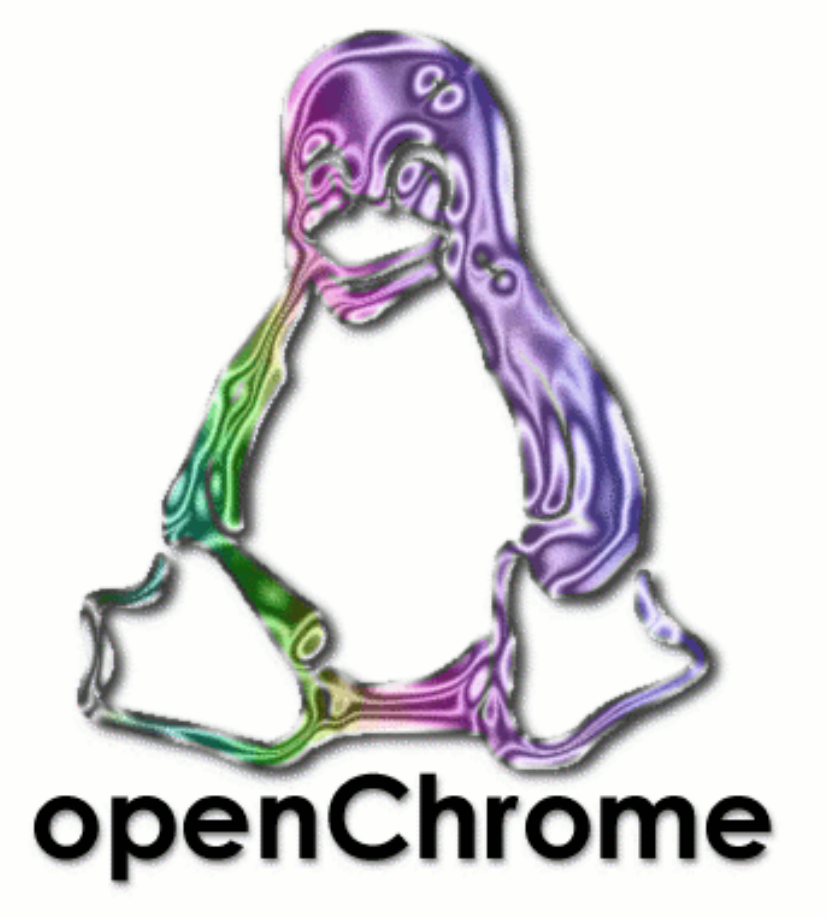
Reviving the Development of OpenChrome



Kevin Brace
OpenChrome Project Maintainer / Developer
XDC2017
September 21st, 2017

Outline

- About Me
- My Personal Story Behind OpenChrome
- Background on VIA Chrome Hardware
- The History of OpenChrome Project
- Past Releases
- Observations about Standby Resume
- Developmental Philosophy
- Developmental Challenges
- Strategies for Further Development
- Future Plans

About Me

- EE (Electrical Engineering) background (B.S.E.E.) who specialized in digital design / computer architecture in college (pretty much the only undergraduate student “still” doing this stuff where I attended college)
- Graduated recently
- First time conference presenter
- Very experienced with Xilinx FPGA (Spartan-II through 7 Series FPGA)
- Fluent in Verilog / VHDL design and verification
- Interest / design experience with external communication interfaces (PCI / PCIe) and external memory interfaces (SDRAM / DDR3 SDRAM)
- Developed a simple DMA engine for PCI I/F validation w/Windows WDM (Windows Driver Model) kernel device driver
- Almost all the knowledge I have is self taught (university engineering classes were not very useful)

Motivations Behind My Work

- General difficulty in obtaining meaningful employment in the digital hardware design field (too many students in the field, difficulty obtaining internship, etc.)
- Collects and repairs abandoned computer hardware (It’s like rescuing puppies!)
- Owns 100+ desktop computers and 20+ laptop computers (mostly abandoned old stuff I collected from various places)
- As an independent newbie developer, working on Big 3 of x86 platform graphics (i.e., Intel, AMD, and NVIDIA) was not an appealing option
- OpenChrome was the least flawed graphics stack outside of the Big 3 after doing lots of testing myself
- Wanted to fix standby resume of an inexpensive VIA chipset based laptop I bought off ebay and then quit!

Ended up Getting Sucked into the Development

- My work with OpenChrome turned into what I will call, “more than part time effort, but not quite full time effort”
- It has been like this since November 2015 or so
- Most of my personal time is spent on OpenChrome development (7 days a week / 52 weeks per year)
- Never been paid for the work
- Generally lost interest in digital hardware design as a result (employability issue played a role)

VIA Chrome Hardware

- Developed by VIA Technologies Taiwan based design team around early 2000s
- (Supposedly) Formed their design team by poaching Taiwan based employees that worked for Trident Microsystems (led to a litigation between two firms)
- (Supposedly) Not related to S3 (later renamed S3 Graphics) Savage line of products
- Only offered as an IGP (Integrated Graphics Processor)
- Targets low end of the market where the die size needs to be minimal (i.e., moderate performance)

VIA Chrome Hardware (cont.)

- Dual Head since Day One (CLE266 chipset)
- Fair amount of design reuse appears to be going on (i.e., register locations are fairly evolutionary)
- Have been ported to many different processor I/F (Intel P6 bus, AMD EV6 bus, AMD HyperTransport, Intel FSB (Pentium 4 / M / Core 2), and VIA V4 bus)
- Two major (?) variants: UniChrome and Chrome9

VIA UniChrome Family

- UniChrome
  (CLE266, KM400(A) / KN400(A), P4M800)
- UniChrome Pro
  (K8M800 / K8N800, P4M800 Pro / VN800 / CN700, PM800 / PN800 / PM880, CN333 / CN400)
- UniChrome Pro II (CX700(M / M2))

VIA UniChrome Family (cont.)

- Generally considered to have Microsoft Direct3D 6 generation 3D hardware support
- (I will assume) 3D functionality / performance is likely comparable to Intel Gen2 IGP
- Marketing (department) appears to emphasize video acceleration features more than 3D functionality / performance
- Video acceleration features varies by device (i.e., CLE266 has a hardware MPEG-2 decoder, but KM400(A) omits it)

VIA Chrome9 Family

- Chrome9 (K8M890 / K8N890)
- Chrome9 HC (P4M900 / VN896 / CN896)
- Chrome9 HC3 (VX800 / VX820)
- Chrome9 HCM (VX855 / VX875)
- Chrome9 HD (VX900 / VX900H)

VIA Chrome9 Family (cont.)

- Major updates were made to the 3D engine
- (I think) The design concept is, “How little die area can we consume in order to obtain Windows Vista Basic / Premium logo from Microsoft?”
- Minimalist Direct3D 9 implementation (comparable to Intel Gen3 IGP like Intel GMA 950)
- (Likely) The general 3D performance is below Intel Gen3 IGP

VIA Chrome9 Family (cont.)

- VIA still supported AMD / Intel platforms until 2007 (K8M890 / K8N890 and P4M900 / VN896 / CN896 chipsets)
- Since VX800 chipset, VIA only supports VIA V4 bus only (V4 bus is merely a slightly modified Intel FSB; a single jumper pin can switch between two processor vendors)
- Since VX800 chipset, VIA became more embedded / netbook focused
- VX855 chipset added HD 1080p hardware acceleration support
- VX900 chipset added integrated HDMI / DP support (VX900H chipset)
- VX900 chipset was the last device (S3 Graphics developed Chrome graphics core took over the VIA IGP socket)

The History of OpenChrome Project

- Initial VIA code dump starting around 2003 / 2004
- Initial code had issues with licensing terms
- VIA did not release hardware programming documents right away (probably not until 2008)
- Since the IGP was called UniChrome, it was originally called The UniChrome Project

The History of OpenChrome Project (cont.)

- Major schism happened sometime around 2005
- One developer (Developer A) wanted to clean up the code and implement VBE (VESA BIOS Extension) free mode setting
- Rest of the developers (Developer B through G) wanted to spend more of their development resources on Xv and XvMC implementation
- Major mailing list shouting match (Is there such a thing?) erupted on xorg-devel mailing list around December 2005 between Developer A and Developer B through G with many core X Server developers appearing to side with Developer B through G

The History of OpenChrome Project (cont.)

- Developer A continued The UniChrome Project mostly alone
- Developer B through G ended up forking The UniChrome Project code and renamed it The OpenChrome Project
- Due to greater personnel resources, OpenChrome ended up having greater device support (i.e., newer devices) than UniChrome
- In the meantime, Developer A also worked on other FOSS projects
- Eventually, pretty much all Linux / BSD distributions ended up adopting OpenChrome instead of UniChrome
- VIA Chrome has had 4 to 5 different graphics stacks just for X / Linux (including VIA closed source device driver stack they tried to get the DRM portion only mainlined back in 2009)

The History of OpenChrome Project (cont.)

- Developer C did a lot of Xv and DMA engine related work, but dropped out around 2009 / 2010
- Developer H joined around late 2010 and worked on OpenChrome until about early 2015
- Developer H got EXA working on OpenChrome (mostly reliable 2D rendering)
- Developer H also started the work on implementing DRI2 / KMS supporting OpenChrome DRM (drm-openchrome) around early 2011
- Developer H effectively abandons the project by Spring 2015
- By mid-2015, the project was effectively abandoned with no coding work occurring
- Developer B was the only one left and he is does not handle active development work (mostly focused on code release)

The History of OpenChrome Project (cont.)

- Purchases my first VIA Chrome IGP based chipset laptop in January 2015 (for OpenChrome development purposes)
- Initially, only interested in fixing standby (ACPI S3 State or Suspend to RAM) resume related bugs
- Originally, I was joking that I might get out of the digital hardware design field if I started to work on this
- Started to learn how to compile DDX around June / July 2015
- Struggled for several months (Quitting for several weeks after getting discouraged and resuming again. Repeated this cycle multiple times. I get to do this presentation since I did not quit.)

The History of OpenChrome Project (cont.)

- Submit first set of patches around November 2015 to freedesktop.org Bugzilla (ignored)
- Was not able to contact Developer C for months (developer H completely ignored my e-mails)
- After embarrassing myself on xorg-devel mailing list around late January 2016, finally obtains OpenChrome repository commit privilege in early February 2016
- Made my first commit on February 9th, 2016

Past OpenChrome DDX Releases

- OpenChrome DDX prior to Version 0.3.3
  - I was not involved in the development
  - Version 0.3 was the first version with UMS / KMS support
- OpenChrome DDX Version 0.4 (March 2016)
  - First official release version I was involved in (first release in 2 ½ years)
  - Removed VBE mode setting
  - Removed the notorious “Known Device Table” (300+ entries), and an X Server crash associated with devices not registered with it (lack of testing by past developers)

Past OpenChrome DDX Releases (cont.)

- OpenChrome DDX Version 0.5 (July 2016)
  - Fixed runtime screen resolution change X Server crash bug (Developer H broke it)
  - Fixed I2C bus 2 (3C5.31) connected display device recognition bug
  - First official dual head support (analog + DVI, analog + FP, etc.)
  - Added VIA Technologies VT1632(A) (Silicon Image SiI 164 clone) support for DVI

Past OpenChrome DDX Releases (cont.)

- OpenChrome DDX Version 0.6 (March 2017)
  - Use of strapping resistors for better automatic display detection (benefits FP and DVI)
  - Fixed a bug that shuts down HP 2133 mini-note's PCIe WLAN (FP / PCIe shared pin or pin multiplexing issue) thanks to the use of trapping resistors
  - Added SiI 164 support for DVI
  - Added official support for CX700 / VX700 / VX800 chipset integrated DVI transmitter
  - First version where I did all the release related work (i.e., including uploading the archived source code to x.org archive) myself without the help of Developer C

The State of OpenChrome DRM

- Work on it started around 2011
- Git repository was initially set up around 2013
- Some of the code appears to originate from VIA Technologies (distinctively different from OpenChrome DDX UMS code)
- Had another developer (Developer I) work on it for several months in 2016, but moved away from the project (due to work commitment)

The State of OpenChrome DRM (cont.)

- I made my first substantive commit around September 2016
- Started to share some code between OpenChrome DDX UMS code (i.e., FP software power on sequence) sometime in 2017
- Due to future OS support issues (3 to 5 years), started to work on updating the Linux kernel used for development from Linux 3.19 to Linux 4.13 in July 2017
- In order to organize the repository better, had to create developmental branches first (finished it around late July)
- Had no prior experience doing this, so figuring out the correct usage of Git took a week in one case (pulling drm-next into the correct branch)
- Sort of copied the branch naming convention of AMDGPU DRM repository (https://cgit.freedesktop.org/~agd5f/linux)

The State of OpenChrome DRM (cont.)

- First, changed the drm directory (from drm/via to drm/openchrome) and module name (from via.ko to openchrome.ko)
- Created drm-next-3.19 branch first as a backup and non-PAE kernel development for VIA C3 processor / CLE266 chipset
- Pulled then latest drm-next (Linux 4.13 rc2 first, 4.13 rc5 a few weeks later) in early August and this became the drm-next-4.13 branch

The State of OpenChrome DRM (cont.)

- Jumping 14 Linux kernel releases was really hard (so many things get discontinued or replaced)
- In order to reduce my “stress” level, used a faster (?) desktop computer (AMD Athlon X2 BE-2300 / K8M890 chipset) to speed up the kernel compilation time
- Full Linux kernel compilation takes roughly 7 hours
- OpenChrome DRM incremental compilation (after the kernel is fully compiled) alone takes about 10 minutes (3 minutes for initial compilation error display)
- AMD Athlon X2 BE-2300 feels like roughly 3 times faster in terms of compile times compared to HP 2133 mini-note’s VIA C7-M 1.2 GHz (productivity issue)

The State of OpenChrome DRM (cont.)

- Backported some drm-next-4.13 changes made to drm-next-3.19 in order to keep the expected DRM functionality coherent
- drm-next-3.19 was working okay (it is still buggy here and there) after the backports
- After clearing up the drm-next-4.13 compilation errors (took about 5 days), it took another 35+ days to properly boot Xubuntu 16.04.3 GUI
- During this dark (?) 35+ days period, tried various things like massive printf debug (DRM_DEBUG_KMS), but did not really help
- Ultimately, comparing the code with Nouveau, Radeon, Matrox G200, and Cirrus Logic QEMU DRMs solved the problem

The State of OpenChrome DRM (cont.)

- A missing ttm_bo_driver struct callback entry (io_mem_pfn) was triggering a null PC (Program Counter) exception, and it was finally traced to a commit made by an AMD developer (will not name names here)
- The AMD developer in question should have put in a null pointer check before calling the callback function (it would have saved me the 35+ days . . .) so it would have left accurate return stack trace information to figure out where the code was crashing
- This commit in question (commit ea642c3) was made sometime during March 2017
- After that fix was made, was able to boot Xubuntu 16.04.3 + Linux 4.13 rc5 (happened on September 13th, 2017, a little over a week ago, just in time for XDC2017)

The State of OpenChrome DRM (cont.)

Issues Remaining:

- Runtime screen resolution / orientation change will crash the X Server (this mostly works on DDX UMS code path)
- Standby resume will lose FP completely (VN896 chipset; can switch to VT on VGA after resume)
- Hardware cursor is missing (K8M890 chipset)
- Still contains “legacy” code from Developer H that doesn’t handle FP mode setting very well due to the lack of trapping resistor use (replace it with proven code modules written by me)
- No support for external DVI (TMDS) transmitter (it is working for DDX UMS code) at this point (I need to learn how to use Linux kernel I2C services)
- CX700 / VX700 chipset integrated DVI (implemented as a DVI-I connector) does not work yet (again I2C bus issue)

Observations about Standby Resume

- Probably the issue I spent most of my development efforts so far (DDX and DRM)
- It is very important not to assume that the video firmware is somehow going to magically reinitialize the hardware resources after standby resume
- It is falsely reassuring that just because the device driver happened to be working correctly at boot time (initial boot), it does not mean that the device driver is properly initializing all the relevant hardware resources (it's called relying on the leftover register values set by someone else)
- However, this assumption will fail miserably when resuming from standby (it is basically your device driver design flaw)
- (True most of the time) OS doesn't really let you know that the system came out of standby, especially to graphics hardware (more about this shortly)

Observations about Standby Resume (cont.)

- The best practices I have developed to cope with this issue is to reinitialize all the relevant hardware resources used by the graphics hardware
- OpenChrome developers historically relied too much on someone else (VGA BIOS) doing this correctly
- Not using hardware strapping (I was the first one to use this somehow) is fatal for FP and DVI reinitialization (i.e., FP data path width determination)
- OpenChrome also uses “hints” from VGA BIOS (it is stored in several scratch pad registers), and this is necessary to handle FP presence (scratch pad registers do not survice standby resume, so it has to be captured by OpenChrome during boot time)
- Assume that the relevant hardware resources like display FIFO parameters, PLL parameters, FP display path, FP display delay tap, etc. are undefined after standby resume (mode setting needs to ALWAYS take care of them)

Observations about OpenChrome DRM Standby Resume

- OpenChrome DRM has had standby resume issue ever since I first tried it last year
- Since I had other pressing issues, I did not really invest a lot of time in fixing the issue (generally being difficult to analyze the issue did discourage me from working on this issue)
- After porting OpenChrome DRM to Linux 4.13 rc5 (September 13th, 2017), I spent some time analyzing why standby resume was failing outright (the system is still running, but the video display is dead; it switches to VT, but no display)

Observations about OpenChrome DRM Standby Resume (cont.)

- After struggling for days, I observed that after coming out of standby resume, OpenChrome DRM did not run through the callback assigned to mode_set member of drm_crtc_helper_funcs
- It was calling the callback assigned to mode_set_base member only
- The problem is, the callback assigned to mode_set_base member does not initialize display FIFO parameters, PLL parameters, and other display controller parameters
- The mode_set member is only called during the initial boot and (probably) runtime screen resolution change (at this time, runtime screen resolution will crash the X Server due to a bug of OpenChrome DRM; probably GEM / TTM related)

Observations about OpenChrome DRM Standby Resume (cont.)

- After reading the drm_crtc_helper_funcs declaration, it turns out that the use of mode_set_base member is optional
- So, I stopped assigning the callback for mode_set_base and only use mode_set member
- This still did not fully solve the issue (X Server still does not reinitialize the display hardware properly), but I can now see something on analog (VGA) output when I switch to VT
- FP is still dead, but the use of VIA Chrome hardware register backdoor manipulation tool can get the FP turned back on (still only in VT)
- This means that if I can retrofit OpenChrome DRM with the proven DDX UMS code for FP initialization, I can probably get FP to display something
- Hopefully, the X Server issue can be solved in the next few months

Development Philosophy

1) Above all, functional correctness and reliability
2) Ease of use (no manual settings)
3) Clean and well organized code
4) Performance (rendering performance)

Development Challenges

- The lack of prior graphics stack development experience
- Did not go through mentoring program like GSoC or EVoC
- There was no work handover from past developers
- Slow (old) computers (It takes roughly 24 hours to compile Linux 4.13 kernel on VIA C7-M 1.2 GHz . .)
- Availability of test samples (i.e., people throwing away their old equipment)
- Equipment deterioration (i.e., secondhand laptops on ebay / resellers are often not in good shape)
- DRM subsystem is a moving target (i.e., constant small changes; virtually every Linux kernel release)

Strategies for Further Development

- Code sharing between OpenChrome DDX UMS code and OpenChrome DRM KMS code
  - Prove the functional correctness in OpenChrome DDX UMS code and then port it to OpenChrome DRM
  - Tremendous compilation time savings (5 min. for DDX vs. 30+ min. for DRM on HP 2133)
  - OpenChrome DDX code is far more mature (stable), so it is a better test target (DRM is still pretty buggy)
- Invest more time in learning various technologies used in the graphics stack (i.e., EXA, GEM / TTM, KMS, etc.)
- Go for hits, not a home run

Future Roadmap – OpenChrome DDX

- OpenChrome DDX Version 0.7 should be released later in the year (November / December 2017)
  - So that it has a chance to go into Ubuntu 18.04 LTS
  - Improved I2C bus display assignment algorithm (for DVI-(I/D) / FP / TV)
  - 24-bit FP I/F FP standby resume bug fix (VN896 chipset)
  - DDX will now search for openchrome.ko, not via.ko for KMS availability

Future Roadmap – OpenChrome DDX (cont.)

- OpenChrome DDX Version 0.8 and beyond will likely have the following improvements
  - Support for VX900 chipset's integrated DVI
  - Backport OpenChrome DRM's working VX900H chipset integrated HDMI code to DDX
  - Support for VX900H chpset's integrated DP (DP registers are public unlike VX900 chipset's HDMI registers)
  - Completely rewrite TV output code for VIA Technologies VT1622 / VT1623 / VT1625 (TV output code is pretty close to broken right now)
  - Further big fixes (i.e., standby resume, display detection, color tone, etc.)
  - KMS only build option (no plans to remove UMS code inside DDX)

Future Roadmap – OpenChrome DRM

- It is currently getting most of my development attention (95% between July 2017 to September 2017)
- Further code modules sharing with OpenChrome DDX UMS (User Mode Setting) code
- Fix standby resume bug (Can now do VT switch with analog, so there is some hope!)
- Fix screen resolution crash bug that was left behind by Developer H (made little progress so far)
- Need to achieve display device support feature parity with OpenChrome DDX UMS code (Analog / int. DVI / ext. DVI / int LVDS FP, ext. LVDS FP; except TV output) for all VIA Chrome devices (CLE266 through VX900 chipsets)
- TV output code will be added later (Again, it is broken already with OpenChrome DDX UMS code.)

Future Roadmap – OpenChrome DRM (cont.)

- Short term goal will be to release unaccelerated, mode setting only DRM without 2D (EXA), video, and 3D acceleration support (hopefully sometime in 2018)
- It will still be using “legacy” KMS code paths (I hope an exception can be granted by the DRM maintainer.)
- Keep up with the latest Linux kernel development so that porting DRM to the new kernel will not be such a pain (the 35+ days thing I just discussed)
- Maintain drm-next-4.1x and drm-next-3.19 branches until the time atomic mode setting needs to be supported (drm-next-3.19 branch will be EOL at that point)

Future Roadmap – Other OpenChrome Graphics Stack Components

- Acceleration support developmental priority will be 2D > 3D > video (in general)
- Device support developmental priority will be Chrome9 (Gallium 3D) > UniChrome (legacy Mesa)
- The motivation for the 3D support is for 3D desktop GUI, hence, Chrome9 will get the most attention
- When the stuff is working well, further development will end (hint for the next slide)

Future Plan – Other Graphics Projects

- I do not mind working on legacy hardware (I am not doing this for a career.)
- I am in for the long haul (Have been doing this for two years straight, and I am just getting started . . .)
- Interested in getting mid to late ‘90s to mid 2000s forgotten graphics devices’ DRM to support DRI2 / KMS / TTM
- Retire DRI1 code by porting it to the new standard
- Not interested in working on Big 3 x86 platform graphics except the really forgotten ones (examples in the next slide)
- Plan to develop the DRM from scratch (of course, I do reference Intel / Nouveau / Radeon DRM for code examples) and reuse the code across many graphics devices

Future Plan – Other Graphics Projects (cont.)

- Forgotten Graphics Hardware Examples:
  - 3Dlabs Glint / Permedia
  - ATI Technologies Mach / RAGE / RAGE 128 (PCI / AGP)
  - Cirrus Logic (PCI / AGP)
  - Matrox Mystique, Millennium, and G series
  - S3 ViRGE / Savage
  - SiS (PCI / AGP / IGP)
  - Trident (PCI / AGP)
  - Other forgotten PCI / AGP generation graphics (Intel 740, Intel 810 / 815, 3dfx, etc.)
- Update relevant DDX / Mesa code to support these devices
- Considering porting all of these DRMs and OpenChrome DRM to one BSD platform (still need 32-bit x86 support)
- All of this will likely take 10+ years at this pace

That’s All

Thank you for listening. Got any questions?